# CSE 1321L: Programming and Problem Solving I Lab

## Lab 3

## Types, Rounding, and Expressions

### What students will learn

- Printing to the screen (i.e. prompting the user)
- Creating variables and assigning values variables
- Reading input from the user and storing it into a variable
- Doing basic calculations with variables to generate a solution

### Content



### Overview:

In this lab, you're going to continue practicing your coding skills by writing programs that interact with the user and do calculations using variables. The labs below also reinforce the concept of creating variables that hold "intermediate solutions" to avoid having one "giant" equation. What you should focus on is trying to understand the problem, understanding the steps needed to solve it, and then converting them into a working program.

Make sure that files are called ***Lab3A.py***, ***Lab3B.py*** and ***Lab3C.py***. Also, please note that while the structure of the output of your program needs to match the samples provided, your solution needs to be correct for any given input.

Lastly, your program output must be exactly the sample output provided for each lab, except for user inputs, which are shown in **red and bold**. This includes any extra blank line, special characters (such as $ or %), prompt and output text, etc. Just follow the sample output.

## Lab3A: *Credit Cards*

Financial advisors will almost always tell you that you should pay for things in cash and avoid credit card debt. Further, they tell you that you should have a small emergency fund that you keep stocked for emergencies like flat tires, dead refrigerators and so on. However, life doesn't always work that way and sometimes we need to charge things for our credit cards. So, for this part of the lab, ***we're going to write a calculator that calculates your minimum monthly payment on your card.***

**For this lab:**

- Write a program that prompts the user for their Current.Balance on their credit card and their Annual.Percentage.Rate.(APR) of the card.
- Make sure to read these inputs as **floats**.
- Then, the program should calculate the Monthly.Percentage.Rate by dividing the APR by 12.
- Use the Monthly Percentage Rate(MPR) to calculate the Minimum.Payment. Remember to use the MPR as a decimal value for this calculation by dividing it by 100.
  You can calculate this value by multiplying the current balance on the credit card (Amount.Owed) times the Monthly.Percentage.Rate:

  Amount Owed × Monthly Percentage Rate = Minimum Payment.

  or

  Amount Owed × APR ÷ 12 = Minimum Payment.

- Lastly, the program should output the Monthly.Percentage.Rate and.Minimum.Payment.

**Note:**

- The input APR is a percentage, so be sure to divide it by 100 when calculating the minimum payment.
- The Monthly Percentage Rate is calculated by dividing the APR by 12 since there are 12 months in a year.
- When printing the monthly percentage rate and the minimum payment, **make sure to round them to 3 and 2 decimal places**, respectively.
- You can round any float by putting it inside round():

```
print(round(3.14159, 2)) #prints PI rounded to two decimal places
```

- Follow the output format shown in the Sample Output. The Monthly Percentage Rate should be inputted as a percentage value, while the Monthly Percentage Rate should be outputted as decimal value, not percentage value.
- **Remember, for calculations, any percentage value must be converted to its decimal form by dividing it by 100.**

Below are two example runs. The user input is shown in **red and bold** (notice the dollar sign is not part of the user input).

**Sample Output #1:**

```
Amount owed: $2000
APR: 19.75
Monthly percentage rate: 1.646
Minimum payment: $32.92
```

**Sample Output #2:**

```
Amount owed: $8500
APR: 29
Monthly percentage rate: 2.417
Minimum payment: $205.42
```

**Sample Output #3:**

```
Amount owed: $5500.25
APR: 37.7
Monthly percentage rate: 3.142
Minimum payment: $172.8
```

## Lab3B: *GPA Calculator*

We're getting more practice making calculators! GPA is important. It's one of the many things that employers look at when recruiting new candidates. You also need a GPA of at least 2.0 to graduate from KSU. GPA is measured by "quality points" using the following scale:

A = 4 quality points
B = 3 quality points
C = 2 quality points
D = 1 quality point
F = 0 quality points

Each course counts for a certain number of credit hours. For instance, most courses are 3 credit hours. This lab is a 1 credit hour course. Calculus counts 4 credit hours. To calculate the quality points for one course, multiply the number of hours of that course with the quality points you earn for that course.

To calculate your GPA for the whole semester, you take the total number of quality points earned that semester and divide it by the total number of hours taken that semester.

**For this lab:**

- Write a program that reads from the user the number of hours and quality.points earned for four courses.
- Make sure to read these inputs as **integers**.
- Then calculates the Total.Hours, Total.Quality.Points, and the GPA.
- Finally, output the Total.Hours, the Total.Quality.Points, and the GPA.

**Note:**

- When printing the total amount of hours and quality points, print them as integers.
- When printing the **GPA,** print it as a **float**, rounded.to.8.decimal.places.

Example runs are shown below. The user input is shown in **red and bold**.

**Sample Output #1:**

```
Course 1 hours: 4
Grade for course 1: 4
Course 2 hours: 3
Grade for course 2: 3
Course 3 hours: 3
Grade for course 3: 4
Course 4 hours: 4
Grade for course 4: 4
Total hours: 14
Total quality points: 53
Your GPA for this semester is 3.79
```

**Sample Output #2:**

```
Course 1 hours: 4
Grade for course 1: 1
Course 2 hours: 1
Grade for course 2: 4
Course 3 hours: 3
```

```
Grade for course 3: 4
Course 4 hours: 3
Grade for course 4: 3
Total hours: 11
Total quality points: 29
Your GPA for this semester is 2.64
```

## Lab3C: *Sandwiches*

We are going to design a program that determines how long an oven at a sandwich shop will take to heat up a sandwich.

**For this lab:**

- The program will prompt the user to enter how many of each sandwich type needs to be cooked.
- Make sure to read these inputs as **integers**.
- It will then print out the number of sandwiches entered for each sandwich type on separate lines.
- Calculate the total amount of time the oven will have to run to cook them all.
- Output the cooking times in minutes and seconds.

Below is a table showing how long each sandwich needs to stay in the oven:

| Sandwich Size | Oven Time |
|---|---|
| Small | 30 Seconds |
| Medium | 60 Seconds |
| Large | 1 Minute and 15 Seconds |
| Extra-Large | 2 Minutes and 15 Seconds |

**Note:**

- The number of minutes and seconds must be printed as integers.

Example runs are shown below. The user input is shown in **red and bold**.

**Sample Output #1:**

```
Enter the number of small sandwiches: 2
Enter the number of medium sandwiches: 2
Enter the number of large sandwiches: 2
Enter the number of extra-large sandwiches: 2

You've entered 2 small sandwiches.
You've entered 2 medium sandwiches.
You've entered 2 large sandwiches.
You've entered 2 extra-large sandwiches.

Total cooking time is 10 minutes and 0 seconds.
```

**Sample Output 2:**

```
Enter the number of small sandwiches: 2
Enter the number of medium sandwiches: 3
Enter the number of large sandwiches: 4
Enter the number of extra-large sandwiches: 5

You've entered 2 small sandwiches.
You've entered 3 medium sandwiches.
You've entered 4 large sandwiches.
You've entered 5 extra-large sandwiches.

Total cooking time is 20 minutes and 15 seconds.
```

## Submission Instructions:

- Programs must follow the output format provided. This includes each blank line, colons (:), and other symbols.
- Programs must be working correctly.
- Programs must be written in Python.
- Programs must be submitted with the correct **.py** format.
- Programs must be saved in files with the correct file name:
  - Lab3A.py
  - Lab3B.py
  - Lab3C.py
- Programs (source code files) must be uploaded to Gradescope by the due date.